(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
2 May 2002 (02.05.2002)

PCT

(10) International Publication Number
WO 02/35380 A1

(51) International Patent Classification⁷: G06F 17/14

(21) International Application Number: PCT/US01/27778

(22) International Filing Date: 23 October 2001 (23.10.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

09/694,452	23 October 2000 (23.10.2000)	US
09/694,448	23 October 2000 (23.10.2000)	US
09/694,455	23 October 2000 (23.10.2000)	US

(71) Applicant: INTERNATIONAL BUSINESS MACHINES CORPORATION [US/US]; New Orchard Road, Armonk, NJ 10504 (US).

(72) Inventors: BRADY, Michael, T.; 1100 E 17th Ave #J201, Longmont, CO 80501 (US). MITCHELL, Joan, L.; 2400 17th Avenue Unit 103D, Longmont, BO 80503 (US). TRELEWICZ, Jennifer, Q.; 1285 E. Imperial Ln., Superior, CO 80027 (US).

(74) Agent: REID, Scott, W.; International Business Machines Corporation, Intellectual Property Law Dept. 9CCA/B002, P.O. Box 12195, Research Triangle Park, NC 27709 (US).

(81) Designated States *(national)*: CN, HU, JP, KR, PL.

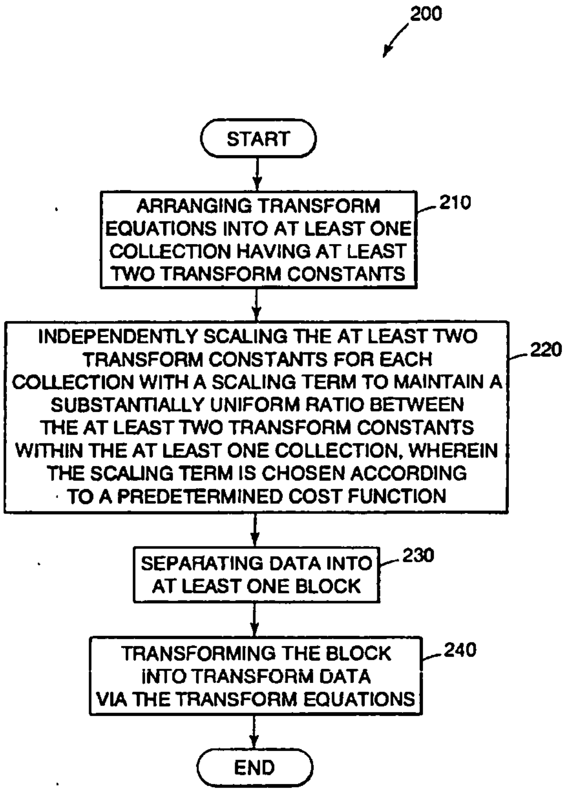
(84) Designated States *(regional)*: European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

Published:

— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: FASTER TRANSFORMS USING SCALED TERMS, EARLY ABORTS, AND PRECISION REFINEMENTS


200
START
ARRANGING TRANSFORM EQUATIONS INTO AT LEAST ONE COLLECTION HAVING AT LEAST TWO TRANSFORM CONSTANTS
210
INDEPENDENTLY SCALING THE AT LEAST TWO TRANSFORM CONSTANTS FOR EACH COLLECTION WITH A SCALING TERM TO MAINTAIN A SUBSTANTIALLY UNIFORM RATIO BETWEEN THE AT LEAST TWO TRANSFORM CONSTANTS WITHIN THE AT LEAST ONE COLLECTION, WHEREIN THE SCALING TERM IS CHOSEN ACCORDING TO A PREDETERMINED COST FUNCTION
220
SEPARATING DATA INTO AT LEAST ONE BLOCK
230
TRANSFORMING THE BLOCK INTO TRANSFORM DATA VIA THE TRANSFORM EQUATIONS
240
END

(57) Abstract: Fast transform (200) that use multiple scaled terms, early aborts, and precision refinements are disclosed. The discrete transforms are split into sub-transforms (210) that are independently calculated using multiple scaling terms on the transform constants (220). The effect of the scaling for the transform coefficients may optionally be handled by appropriately scaling the quantization values or any comparison values. The possible need to perform a corrective action is detected based upon testing the incremental calculations of transform coefficients. Corrective action is then performed. The corrective action includes refining the incremental calculations to obtain additional precision and/or aborting the incremental calculations when the resulting numbers are sufficient.

FASTER TRANSFORMS USING SCALED TERMS, EARLY ABORTS, AND PRECISION REFINEMENTS

BACKGROUND OF THE INVENTION

1. Field of the Invention.

This invention relates in general to data processing, and more particularly to faster transforms that use multiple scaled terms, early aborts, and precision refinements.

2. Description of Related Art.

Transforms, which take data from one domain (e.g., sampled data) to another (e.g., frequency space), are used in many signal and/or image processing applications. Such transforms are used for a variety of applications, including, but not limited to data analysis, feature identification and/or extraction, signal correlation, data compression, and data embedding/watermarking.

In data processing, data is typically represented as a sampled discrete function. The discrete representation is either made deterministically or statistically. In a deterministic representation, the point properties of the data are considered, whereas, in a statistical representation, the average properties of the data are specified. For example, it is well-known in the art to use a discrete cosine transform for data compression. In particular examples referred to herein, the terms images and image processing will be used. However, those skilled in the art will recognize that the present invention is not meant to be limited to processing images but is applicable to processing different data, such as audio data, scientific data, image data, etc.

In a digital image processing system, digital image signals are formed by first dividing a two-dimensional image into a grid. Each picture element, or pixel, in the grid has associated therewith a number of visual characteristics, such as brightness and color. These characteristics are converted into numeric form. The digital image signal is then formed by assembling the numbers associated with each pixel in the image into a sequence which can be interpreted by a receiver of the digital image signal.

Data compression is desirable in many data handling processes, where too much data is present for practical applications using the data. Commonly, compression is used in communication links, to reduce transmission time or required bandwidth. Similarly, compression is preferred in image storage systems, including digital printers and copiers, where "pages" of a document to be printed may be stored temporarily in memory. Here the amount of media space on which the image data is stored can be substantially reduced with compression. Generally speaking, scanned images, i.e., electronic representations of hard copy documents, are often large, and thus make desirable candidates for compression.

Signal and image processing frequently require converting the input data into transform coefficients for the purposes of analysis. Often only a quantized version of the coefficients is needed (e.g. JPEG/MPEG data compression or audio/voice compression). Many such applications need to be done fast in real time such as the generation of JPEG data for high speed printers. Many of these transforms require efficient implementation for real-time and/or fast execution whether or not compression is used as part of the data processing.

Pressure is on the data signal processing industry to find the fastest method by which to most effectively and quickly perform the digital signal processing. As in the field of compression generally, research is highly active and competitive in the field of fast transform implementation. Researchers have made a wide variety of attempts to exploit the strengths of the hardware intended to implement the transforms by exploiting properties found in the transform and inverse transform.

One such technique is the ISO 10918-1 JPEG International Standard /ITU-T Recommendation T.81. The draft JPEG standard is reproduced in Pennebaker and Mitchell, JPEG: *Still Image Data Compression Standard*, New York, Van Nostrand Reinhold, 1993, incorporated herein by reference. One compression method defined in the JPEG standard, as well as other emerging compression standards, is discrete cosine transform (DCT) coding. Images compressed using DCT coding are decompressed using an inverse transform known as the inverse DCT (IDCT). An excellent general reference on DCTs is Rao and Yip, *Discrete Cosine Transform*, New York, Academic Press, 1990, incorporated herein by reference. It will be assumed that those of ordinary skill in this art are familiar with the contents of the above-referenced books.

It is readily apparent that if still images present storage problems for computer users and others, motion picture storage problems are far more severe, because full-motion video may require up to 60 images for each second of displayed motion pictures. Therefore, motion picture compression techniques have been the subject of yet further development and standardization activity. Two important standards are ISO 11172 MPEG International Standard and ITU-T Recommendation H.261. Both of these standards rely in part on DCT coding and IDCT decoding.

The two-dimensional discrete cosine transform is a pair of mathematical equations that transforms one N_1xN_2 array of numbers to or from another N_1xN_2 array of numbers. The first array typically represents a square NxN array of spatially determined component values which form the digital image. Since each pixel may have associated with it several components, in what follows, a single-component pixel is considered for simplicity. However, the extension to multi-component pixels is obvious to practitioners in the art. The second array is an array of discrete cosine transform coefficients which represent the image in the frequency domain. This method of representing the image by the coefficients of its frequency components is a special case of the discrete Fourier transform. The discrete Fourier transform is the discrete version of the classic mathematical Fourier transform wherein any periodic waveform may be expressed as a sum of sine and cosine waves of different frequencies and amplitudes. The discrete cosine transform, like the Fourier transform, is thus a transform which transforms a signal from the time domain into the frequency domain and vice versa. All DCT multiplications are real. This lowers the number of required multiplications, as compared to the discrete Fourier transform. For most images, much of the signal energy lies at low frequencies; these appear in the upper left corner of the DCT. The lower right values represent higher frequencies, and are often small enough to be neglected with little visible distortion.

There are two basic discrete cosine transform equations. The first basic equation is the forward discrete cosine transform which transforms the pixel values into the discrete cosine transform coefficients. The second basic equation is the inverse discrete cosine transform which transforms the discrete cosine transform coefficients back into pixel values. Most applications of the discrete cosine transform for images use eight-by-eight arrays wherein N therefore has a value of eight. Assuming then that N has the value of eight when performing the transforms, where f(i, j) are the values of the pixel array and F(u, v) are the values of the discrete cosine transform coefficients, the formula for the 2D discrete cosine transform is given by:

$$F(u,v)=\frac{C_u C_v}{4}\sum_{x=0}^{7}\sum_{y=0}^{7}f(x,y)\cos\left(\frac{(2x+1)u\pi}{16}\right)\cos\left(\frac{(2y+1)v\pi}{16}\right)$$

where x, y = spatial coordinates in the spatial domain (0, 1, 2, ...7); u, v = coordinates in the transform domain (0, 1, 2, ...7); $C_u = \frac{1}{\sqrt{2}}$ for u = 0, otherwise 1; and $C_v = \frac{1}{\sqrt{2}}$ for v = 0, otherwise 1. The separable nature of the 2D DCT is exploited by performing a 1D DCT on the eight columns, and then a 1D DCT on the eight rows of the result. Several fast algorithms are available to calculate the 8-point 1D DCT.

As described above, the DCT portion of a compressor comprises mainly two parts. The first part transforms highly correlated image data into weakly correlated coefficients using a DCT transform and the second part performs quantization on coefficients to reduce the bit rate for transmission or storage. However, the computational burden in performing a DCT is demanding. For example, to process a one-dimensional DCT of length 8 pixels requires 13 multiplications and 29 additions in currently known fast algorithms. (In the article, *A Fast DCT-SQ Scheme for Images*, Trans. IEICE, Vol. E-71, No. 11, pp. 1095-1097, Nov. 1988, Y. Arai, T. Agui, and M. Nakajima proposed that many of the DCT multiplications can be formulated as scaling multipliers to the DCT coefficients. The DCT after the multipliers are factored out is called the scaled DCT. The scaled DCT is still orthogonal but no longer normalized, whereas the scaling factors may be restored in a subsequent quantization process. Arai, et al. have demonstrated in their article that only 5 multiplications and 29 additions are required in processing an 8-point scaled DCT.) As stated above, the image is divided into square blocks of size 8 by 8 pixels, 16 by 16 pixels or 32 by 32 pixels. Each block is often processed by the one-dimensional DCT in row-by-row fashion followed by column-by-column. On the other hand, different block sizes are selected for compression due to different types of input and different quality requirements on the decompressed data.

Scaled terms may be used to replace multiplicative constants like cosine terms in a Discrete Cosine Transform (DCT) with a minimum number of additions/subtractions. However, the scaled terms merely approximate the constants in the transform equations. Thus, some error is accepted to keep the precision confined to a fixed number of bits or to minimize the number of operations. If the resulting numbers are further from a decision boundary (e.g., a threshold value or a quantization boundary) than the maximum possible error, the result will not be affected by the approximations. However, the resulting numbers may be determined, during the incremental calculations, to require additional precision. Yet, the original input values are no longer available in the registers, and refetching the original input values from memory can impose cycles associated with cache misses and memory latency. The brute-force option is to perform an inverse transform (e.g., an IDCT) on the values, and then re-run the forward transform (e.g., FDCT, sometimes denoted just DCT) with higher precision. The disadvantage of the brute force approach is that operations are wasted.

However, research generally focuses on specific techniques, such as the above-mentioned techniques that used DCT coding to provide the desired degree of compression. Nevertheless, other transforms may be used to provide certain advantages under certain circumstances. For example, in the DCT compression coding method discussed above, an input image is divided into many uniform blocks and the two-dimensional discrete cosine transform function is applied to each block to transform the data samples into a set of transform coefficients to remove the spatial redundancy. However, even though a high compression rate may be attained, a blocking effect, which may be subtle or obvious, is generated. Further, vector quantization methods that may be utilized by the compression system are advantageous due to their contribution to the high compression rate. On the other hand, a sub-band method may reduce the blocking effect which occurs during high rates of data compression. The discrete wavelet transform (DWT) or Sub-Band Coding (SBC) methods encode signals based on, for example, time and frequency components. As such, these transform methods can be useful for analyzing non-stationary signals and have the advantage that they may be designed to take into account the characteristics of the human visual system (HVS) for image analysis.

It can be seen then that there is a need to provide a method and apparatus that provides faster transform calculations, using early aborts and precision refinements to save processing cycles, thus providing decreased software execution times and reduced hardware requirements.

SUMMARY OF THE INVENTION

To overcome the limitations in the prior art described above, and to overcome other limitations that will become apparent upon reading and understanding the present specification, the present invention discloses fast transforms that use multiple scaled terms, early aborts, and precision refinements.

The present invention solves the above-described problems by splitting discrete transforms into sub-transforms that are independently calculated using scaled terms on the transform constants. Further, optimal representations of the scaled terms for binary arithmetic are found. The resulting calculations result in fast transform calculations, decreased software execution times and reduced hardware requirements. Moreover, those skilled in the art will recognize that the inverse transform can often be implemented using the same method so that, in general, the same number of operations is used. The present invention also solves the above-described problems by decreasing the number of multiplications by using scaling terms on the transform constants. Further, after scaling, the terms of the constants are replaced by a series of fewer linear shifts and additions since the constants may be approximated by sums of powers-of-2 after only a few summations. Those skilled in the art will recognize that throughout this specification, the term 'matrix" is used in both its traditional mathematical sense and also to cover all hardware and software systems which when analyzed could be equivalently represented as a mathematical matrix. The present invention also solves the above-described problems by detecting when to perform a corrective action based upon testing the incremental calculations of transform constants and performing the corrective action: refining the incremental calculations to obtain additional precision and/or aborting the incremental calculations when the resulting number is going to be too small.

A method in accordance with the principles of the present invention includes arranging transform equations into at least one collection having at least two transform constants and independently scaling the at least two transform constants for each collection with a scaling term to maintain a substantially uniform ratio between the at least two transform constants within the at least one collection. The method also involves representing each of the scaled discrete cosine transform constants with estimated scaled discrete cosine transform constants approximated by sums of powers-of-2.

Other embodiments of a method in accordance with the principles of the invention may include alternative or optional additional aspects. One such aspect of the present invention is that the method further includes separating data into at least one block and transforming the block into transform data via the scaled transform equations.

Another aspect of the present invention is transforming the block into transformed data by performing matrix multiplication on the discrete cosine transform equations based upon binary arithmetic using the estimated scaled discrete cosine transform constants and performing linear shifts and additions.

Another aspect of the present invention is that the scaling term is chosen according to a predetermined cost function.

Another aspect of the present invention is that the predetermined cost function comprises selecting the scaling term so that the largest error on any transform coefficient is no larger than a predetermined error percentage.

Another aspect of the present invention is that the predetermined cost function comprises selecting the scaling term so that the largest error on each involved transform coefficient is no larger than its individual predetermined error percentage.

Another aspect of the present invention is that the predetermined cost function comprises selecting the scaling term so that predetermined transform constants have an error less than or equal to a predetermined error percentage.

Another aspect of the present invention is that the predetermined cost function comprises selecting the scaling term so that each involved predetermined transform constant has an error less than or equal to its individual predetermined error percentage.

Another aspect of the present invention is that the predetermined cost function comprises selecting the scaling term and representations for the transform constants so that all transform constants for a collection possess simultaneous binary representations with predetermined characteristics.

Another aspect of the present invention is that the cost function minimizes a number of add operations.

Another aspect of the present invention is that the cost function minimizes a worst case number of add operations.

Another aspect of the present invention is that the predetermined characteristics comprise a minimum number of common power-of-2 terms.

Another aspect of the present invention is that the selecting of the scaling term and representations for the transform constants so that all transform constants for a collection possess simultaneous binary representations with a minimum number of common power-of-2 terms is implemented when binary arithmetic shifts may be more efficient than multiplication operations.

Another aspect of the present invention is that the predetermined characteristics comprise a maximized clustering of non-zero power-of-2 terms.

Another aspect of the present invention is that the selecting of the scaling term so that all transform constants for a collection possess simultaneous binary representations with a maximized clustering of non-zero power-of-2 terms is implemented when multiplication operations employing smaller integers are more desirable than multiplies employing larger numbers.

Another aspect of the present invention is that whether the coefficient in a power-of-2 polynomial representing the constant is non-zero is tracked.

Another aspect of the present invention is that a value of the bit position determines the power-of-2 term.

Another aspect of the present invention is that maximizing the clustering of non- zero power-of-2 terms includes finding all representations of the scaled constants by a) setting a first variable to an ith element in the block, b) initializing a second variable to a value of 2, c) initializing a bitmask to binary 3, d) analyzing the bits to determine whether the ith element indicated by the first variable is a candidate representation for doing the term reordering using 2^n +

$2^{n-1} = 2^{n+1} - 2^{n-1}$, e) encoding the ith element by adding the second variable to the first variable to perform an effective power-of-2 change given by $2^n + 2^{n-1} = 2^{n+1} - 2^{n-1}$, f) obtaining a new representation and incrementing the first variable to the i+1th element, g) shifting the mask and second variable left one bit and h) repeating d-g.

Another aspect of the present invention is that the method further includes shifting the mask left after checking if the first variable matching the mask bits were set thereby putting a zero at the right and increasing the power of 2 that is used for reordering in $2^n + 2^{n-1} = 2^{n+1} - 2^{n-1}$.

Another aspect of the present invention is that the collections represent disjoint sets of transform equations of partial calculations.

Another aspect of the present invention is that the collections do not represent disjoint sets of transform equations of partial calculations.

Another aspect of the present invention is that the method further includes selecting an independent scaling term for the transform constants in each of the at least one collections.

A method in accordance with the principles of the present invention includes testing at least one number resulting from an incremental calculation of transform coefficients during a transform, determining whether to perform a corrective action based upon the testing and performing the corrective action when a corrective action is determined to be needed.

Other embodiments of a method in accordance with the principles of the invention may include alternative or optional additional aspects. One such aspect of the present invention is that the determining comprises detecting whether the incremental calculation of the transform coefficients will result in transform coefficients with unacceptable precision and the performing corrective action comprises refining the at least one number.

Another aspect of the present invention is that the transform comprises a transform matrix and wherein the refining comprises applying a refinement matrix for increasing precision of the incremental calculation of the transform constants.

Another aspect of the present invention is that the refinement matrix comprises $I +_d D_{m+1} D_m^{-1}$.

Another aspect of the present invention is that the method further includes generating at least one refinement matrix based on approximately calculated transform constants.

Another aspect of the present invention is that the generating at least one refinement matrix is performed offline or at initialization.

Another aspect of the present invention is that the generating the refinement matrix comprises recognizing that an approximate transform is invertible, generating the refinement matrix given by $I +_d D_{m+1} D_m^{-1}$, and structuring the transform for efficient computation.

Another aspect of the present invention is that the generating the refinement matrix includes recognizing that recovery of the nth column of a transform matrix for generating the transform is impossible, calculating a pseudo inverse for a portion of the transform matrix and generating an approximation for the refinement matrix using the pseudo inverse for the transform matrix.

Another aspect of the present invention is that the approximation of the refinement matrix comprises $I +_d D_1 {}_d \tilde{D}_0$.

Another aspect of the present invention is that the determining further comprises determining whether an error resulting from terminating the incremental calculation is acceptable and the performing corrective action comprises aborting the incremental calculation of a transform coefficient.

Another aspect of the present invention is that the incremental calculation is terminated when a determination is made that the incremental calculation will result in a number that is projected to be within a predetermined range.

Another aspect of the present invention is that the number that is projected to be within a predetermined range comprises a transform coefficient that does satisfy a precision requirement.

Another aspect of the present invention is that the incremental calculation is terminated when a refinement to the transform coefficient is determined not to change the result.

Another aspect of the present invention is that a refinement to the transform coefficient is determined not to change the result when, after checking the relative magnitudes of the results of the incremental calculations, an intermediate calculation of at least one transform coefficient is small compared to the intermediate calculation of another transform coefficient.

Another aspect of the present invention is that a refinement to the transform coefficient is determined not to change the result when, after checking the magnitude of the results of at least one incremental calculation, at least one intermediate calculation of the transform coefficient is less than a predetermined threshold.

Another aspect of the present invention is that the determining further comprises determining that a transform coefficient is going to be within a predetermined range of zero and the performing corrective action comprises aborting the incremental calculation of the transform coefficient.

In another embodiment of the present invention, a data compression system is provided. The data compression system includes a transformer for applying a linear transform to decorrelate data into transform coefficients using transform equations, the transform equations being formed by arranging transform equations into at least one collection having at least two transform constants and independently scaling the at least two transform constants for each collection with a scaling term to maintain a substantially uniform ratio between the at least two transform constants within the at least one collection, wherein the scaling term is chosen according to a predetermined cost function and a quantizer for quantizing the transformed data into quantized data by reducing a number of bits needed to represent the transform coefficients.

In another embodiment of the present invention, a printer is provided. The printer includes memory for storing image data, a processor for processing the image data to provide a compressed print stream output and a printhead driving circuit for controlling a printhead to generate a printout of the image data, wherein the processor applies a linear transform to decorrelate data into transform coefficients using transform equations, the transform equations being formed by arranging transform equations into at least one collection having at least two transform constants and independently scaling the at least two transform constants for each collection with a scaling term to maintain a substantially uniform ratio between the at least two transform constants within the at least one collection, wherein the scaling term is chosen according to a predetermined cost function

In another embodiment of the present invention, an article of manufacture is provided. The article of manufacture includes a program storage medium readable by a computer, the medium tangibly embodying one or more programs of instructions executable by the computer to perform a method for arranging transform equations into at least one collection having at least two transform constants and independently scaling the at least two transform constants for each collection with a scaling term to maintain a substantially uniform ratio between the at least two transform constants within the at least one collection, wherein the scaling term is chosen according to a predetermined cost function.

In another embodiment of the present invention, a data analysis system is provided. The data analysis system includes transform equations being formed by arranging transform equations into at least one collection having at least two transform constants and independently scaling the at least two transform constants for each collection with a scaling term to maintain a substantially uniform ratio between the at least two transform constants within the at least one collection, wherein the scaling term is chosen according to a predetermined cost function and a transformer for applying the transform equations to perform a linear transform to decorrelate data into transform coefficients.

These and various other advantages and features of novelty which characterize the invention are pointed out with particularity in the claims annexed hereto and form a part hereof. However, for a better understanding of the invention, its advantages, and the objects obtained by its use, reference should be made to the drawings which form a further part hereof, and to accompanying descriptive matter, in which there are illustrated and described specific examples of an apparatus in accordance with the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

Referring now to the drawings in which like reference numbers represent corresponding parts throughout:

Fig. 1 illustrates a typical image compression system;

Fig. 2 illustrates a block diagram of a JPEG encoder;

Fig. 3 illustrates a flow chart of the present invention;

Fig. 4 illustrates a flow chart of a method for finding the simultaneous representations for the first criteria, C1, or second criteria, C2;

Fig. 5 illustrates a flow chart for providing faster transforms using corrective action to provide faster transform calculations and decreased execution times;

Fig. 6 illustrates a flow chart of the abort method according to the present invention that demonstrates aborting further iterations of the transform coefficient calculation process;

Fig. 7 is illustrates the testing of the at least one incrementally calculated number;

Fig. 8 is a flow chart of the refinement method according to the present invention;

Fig. 9 illustrates a flow chart of a first method for generating a refinement matrix;

Fig. 10 is a flow chart showing a second method for generating a refinement matrix when ${}_dD_0$ is not invertible;

Fig. 11 illustrates a printer according to the present invention;

Fig. 12 illustrates a data analyzing system according to the present invention; and

Fig. 13 illustrates another data analyzing system according to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

In the following description of the exemplary embodiment, reference is made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration the specific embodiment in which the invention may be practiced. It is to be understood that other embodiments may be utilized as structural changes may be made without departing from the scope of the present invention.

The present invention provides fast transforms that use multiple scaled terms. The transforms are split into sub-transforms that are independently calculated using scaled terms for the transform constants. Optimal representations of the scaled terms for binary arithmetic are found. This method results in fast transform calculations, decreased software execution times and reduced hardware requirements. According to the present invention, discrete transforms used in signal and image processing employ what are called basis functions which set the structure of the transform and form the grounds for allowing the transform to be calculated in two or more sub-collections. Cost functions for fast implementations of the transforms are then used to find optimal representations of the basis coefficients in the calculation.

Fig. 1 illustrates a typical image compression system 100. The data compression system may include three closely connected components namely (a) Transform 120, (b) Quantizer 130, and (c) Optional Entropy Encoder 140. Compression is accomplished by applying a linear transform to decorrelate the image data 110, quantizing the resulting transform coefficients, and, if desired, entropy coding the quantized values. A variety of linear transforms have been developed which include Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT) and many more, each with its own advantages and disadvantages.

The quantizer 130 simply reduces the number of bits needed to store the transformed coefficients by reducing the precision of those values. Since this is a many-to-one mapping, it is a lossy process and is a significant source of compression in an encoder. Quantization can be performed on each individual coefficient, which is known as Scalar Quantization (SQ). Quantization can also be performed on a collection of coefficients together, and this is known as Vector Quantization (VQ). Both uniform and non-uniform quantizers can be used depending on the problem at hand.

The optional entropy encoder 140 further compresses the quantized values losslessly to give better overall compression. It uses a model to accurately determine the probabilities for each quantized value and produces an appropriate code based on these probabilities so that the resultant output code stream will be smaller than the input stream. The most commonly used entropy encoders are the Huffman encoder and the arithmetic encoder, although for applications requiring fast execution, simple run-length encoding (RLE) has proven very effective.

The term image transforms usually refers to a class of unitary matrices used for representing images. This means that images can be converted to an alternate representation using these matrices. These transforms form the basis of transform coding. Transform coding is a process in which the coefficients from a transform are coded for transmission.

The signal (x) is a function mapping each integer from 0..n - 1 into a complex number. An example is given by a line of a sampled or pixelated image, where the samples or pixels are equally spaced. An "orthogonal basis" for a collection of such (x) is a set

$\{b_y(x)\}_{y=0}^{n-1}$ of functions, where $\sum_{x=0}^{n-1} b_y(x)b_z(x)$ = 0 for y ¹ z. A "transform" of (x), denoted F(y), is given by F(y) = $\sum_{x=0}^{n-1} f(x)b_y(x)$. Transforms of this type are used in many signal and image processing applications to extract information from the original signal . One example of a transform is the discrete Fourier transform (DFT), where b_y(x) = exp (2pixy/n). A related example is the discrete cosine transform (DCT), where b_y(x) = cos (2pxy/n). Another example is the wavelet transform, where b_y(x) is a particular scaled and offset version of the mother wavelet function. (See, Ingrid Daubechies, *Ten Lectures on Wavelets*, Society for Industrial & Applied Mathematics, (May 1992)).

The theoretical basis for the independent scaling operations will now be demonstrated by showing the mathematical basis for being able to perform the scales without destroying the structure of the transform. Define a transform F(y) = $\sum_{x=0}^{n-1} f(x)b_y(x)$. Consider those cases (described below) when the b_y(x) are such that this transform can be split into two or more disjoint sums, regardless of the structure of (x). (The term "disjoint", when used herein in reference to the sets of equations, means that there are no transform coefficients in common between equations in the two disjoint sets of equations.) For example, if b_{2y}(x) have even symmetry, and b_{2y+1}(x) have odd symmetry, it is known from mathematics that any (x) can be written uniquely as (x) = $_e$(x) + $_o$(x), where $_e$(x) is even (symmetric about zero) and $_o$(x) is odd (anti-symmetric about zero), and that $\sum_x f_e(x)b_{2y-1}(x) = \sum_x f_o b_{2y}(x) = 0$. This enables the transform to be written equivalently as:

$$F(y) = \sum_{y=0}^{\lfloor (n-1)/2 \rfloor} f_e(x)b_{2y}(x) + \sum_{y=1}^{\lfloor n/2 \rfloor} f_o(x)b_{2y-1}(x)$$

Fig. 2 illustrates a block diagram of a JPEG encoder 100. In Fig. 2, digital image data 110 is divided up into 8 by 8 pixel blocks 112. Then, the discrete cosine transform (DCT) of each block is calculated 120. The discrete cosine transform (DCT) helps separate the data into parts (or spectral sub-bands) of differing importance (e.g., with respect to an image's visual quality). The DCT is similar to the discrete Fourier transform: it transforms a signal from the spatial domain to the frequency domain.

A quantizer 130 rounds off the DCT coefficients according to the quantization matrix. This step produces the "lossy" nature of JPEG, but allows for large compression ratios. There is a tradeoff between image quality and degree of quantization. A large quantization step size can produce unacceptably large image distortion. This effect is similar to quantizing Fourier series coefficients too coarsely; large distortion would result. Unfortunately, finer quantization leads to lower compression ratios. The question is how to quantize the DCT coefficients most efficiently. Because of human eyesight's natural high frequency roll-off, these frequencies play a less important role than low frequencies. This lets JPEG use a much higher step size for the high frequency coefficients, with little noticeable image deterioration. The quantizer coefficient output may then be encoded by an optional entropy encoder 140 to produce a compressed data stream 150 to an output file. Note that for JPEG the entropy encoder is not optional, but other similar data compression systems could be designed without the CPU cycles required by the entropy encoder.

However, as described below, there is a need to provide a method and apparatus for performing discrete cosine transforms with less multiplication steps to increase throughput of the encoder 100. As will be described, the method according to the present invention saves multiplications in the brute force equations by scaling the coefficient matrix. Each separable subgroup is scaled independently of the other collections. Within each collection, the remaining multiplications are replaced by simple shifts and adds. The scaling terms are chosen according to various cost functions. The preferred embodiment uses cost functions that minimize the number of adds and that minimize the worst case number of adds. However, those skilled in the art will recognize that alternate cost functions could choose how much error is allowed per coefficient. Moreover, those skilled in the art will recognize that the inverse DCT (IDCT) can be implemented using the same method so that the same number of operations are used.

Fig. 3 illustrates a flow chart 200 of the present invention. In Fig. 3, transform equations are split into at least one sub-transform having at least two transform constants 210. The term "sub-transforms", as used herein, references the collection of equations used to generate a subset of the transformed terms, where the subset may contain all of the transformed terms, or fewer that the total number of transformed terms. Next, the transform constants for each collection are independently scaled with a scaling term to maintain a substantially uniform ratio between the transform constants within the collection, wherein the scaling term is chosen according to a predetermined cost function 220. The result is the transform equations for transforming the block. The data is separated into at least one block 230. The block is then transformed into transformed data using the transform equations 240. Traditional systems would perform blocks 210 and 220 offline. However, for algorithms in which are specified at run time such as JPEG 2000, these blocks would be automatically done during the program initialization phase. Referring to the quantizer 130 of Fig. 1, the transformed data may then be quantized by incorporating the scaling into the quantization. This is in contrast to the Integer Cosine Transform (ICT) (see, for example, Pang et al., *A self-timed chip for image coding*, IEEE Trans. Circuits and Systems for Video Tech., vol. 9, no. 6, 1999, pp. 856-860.), where each coefficient is scaled by the same value in every equation, a method which does not provide the same computational advantages as the method described herein.

Choosing the scaled term for the constants may be performed with the use of a cost function that represents the needs of the target system. Several parameters to this cost function were identified for fast transforms. The actual cost function that is used may include one or a combination of the following cost functions:

The first cost function is where the largest error on any transform coefficient (i.e., F(x)) is no larger than P%, where P is selected by the algorithm designer. In the example below, P = 1.

The second cost function is where some more important transform constants, as selected by the algorithm designer, have no more than $P_{important}$% error. In the example below, the low frequency terms are less than 0.1% in error.

The third cost function is split into two criteria, C1 and C2, which have applications to different systems; i.e., one generally chooses to satisfy either criterion C1 or criterion C2. Criterion C1 is applicable to implementations (e.g., software or firmware) where shifts may be more efficient than multiplies. Criterion C2 is applicable to implementations where multiplies employing smaller integers are more desirable than multiplies employing larger numbers (e.g., software implementations where multiplies above a certain number of bits use more cycles, or where total precision is a concern.) Specific examples are given to illustrate these two criteria. Both criteria could be used together.

According to the first criterion, C1, all scaled integer basis multipliers (in the case of the FDCT, the constants C_1 through C_7, where C_k = cos(kp/16)) appearing in the same sum should possess simultaneous binary representations with the minimum number of common power-of-2 terms, i.e., the set of power-of-2 terms over all of the representations within a collection is as small as possible. The real-number constants may be approximated by rational numbers; i.e., ratios of integers. The word "representation", as used herein in reference to the scaled constants of the transform, refers to the way in which the numerator of this ratio of integers may be calculated as sums and/or differences of powers-of-two.

For example, in the FDCT, the transform can be broken into three sets of equations as shown:

$$2S(0) = C_4(s_{0734} + s_{1625}) \qquad (1)$$

$$2S(4) = C_4(s_{0734} - s_{1625})$$

$$2S(2) = C_2 d_{0734} + C_6 d_{1625} \qquad (2)$$

$$2S(6) = C_6 d_{0734} - C_2 d_{1625}$$

$$2S(1) = C_1 d_{07} + C_3 d_{16} + C_5 d_{25} + C_7 d_{34} \qquad (3)$$

$$2S(3) = C_3 d_{07} - C_7 d_{16} - C_1 d_{25} - C_5 d_{34}$$

$$2S(5) = C_5 d_{07} - C_1 d_{16} + C_7 d_{25} + C_3 d_{34}$$

$$2S(7) = C_7 d_{07} - C_5 d_{16} + C_3 d_{25} - C_1 d_{34}$$

The notation used for these equations is taken from the Pennebaker book. Note that the Ss are proportional to the $\hat{F}s$. Thus, the constants in set 3 should have simultaneous binary representations with the minimum number of common power-of-2 terms, but their simultaneous representations are not relevant to those used for the constants in set 2. A specific example for set 3 is given by the following:

$$41C_1 \approx 40 = 2^5 + 2^3$$

$$41C_3 \approx 34 = 2^5 + 2^1$$

$$41C_5 \approx 23 = 2^5 - 2^3 - 2^1 + 2^0$$

$$41C_7 \approx 2^3$$

All of these representations have less than 1% error per coefficient. In this example, the set of common power-of-2 terms is $\{2^5, 2^3, 2^1, 2^0\}$, as can be seen from the equations. These representations may be viewed as polynomials in 2 with +1 or 0 multiplying each power of 2; i.e.,

$$4C_1 \approx 1 \cdot 2^5 + 1 \cdot 2^3 + 0 \cdot 2^1 + 0 \cdot 2^0 \qquad (4)$$

What makes minimizing the number of these terms efficient is the following: the set 3 equations can be calculated by grouping these powers of 2. In the architecture that this criterion C1 is used for, adds and shifts are cheaper in cycles than multiplies, so we can look at the set 3 equations as calculating the matrix operation

$$\begin{bmatrix} 2^5+2^3 & 2^5+2^1 & 2^5-2^3-2^1+2^0 & 2^3 \\ 2^5+2^1 & -2^3 & -(2^5+2^3) & -(2^5-2^3-2^1+2^0) \\ 2^5-2^3-2^1+2^0 & -(2^5+2^3) & 2^3 & 2^5+2^1 \\ 2^3 & -(2^5-2^3-2^1+2^0) & 2^5+2^1 & -(2^5+2^3) \end{bmatrix} \begin{bmatrix} d_{07} \\ d_{16} \\ d_{25} \\ d_{34} \end{bmatrix}$$

which takes a total of 28 operations, counting adds and shifts. If we precalculate A = $d_{25} - d_{34}$ and B= d_{07} + d_{16}, the equation becomes

$$\begin{bmatrix} -d_{25} & d_{16} & d_{07}-A & d_{25}+B \\ d_{34} & d_{07} & -d_{16}-A & d_{07}-d_{25}-d_{34} \\ -d_{07} & d_{34} & d_{25}-B & d_{34}+d_{07}-d_{16} \\ d_{16} & d_{25} & -d_{34}+B & -d_{16}+A \end{bmatrix} \begin{bmatrix} 2^0 \\ 2^1 \\ 2^3 \\ 2^5 \end{bmatrix}$$

which, with the precalculations, comes to 24 cycles. Furthermore, eliminating the 2^0 term, which raises the error on C_5 to about 3%, reduces the total number of operations to 20. It should be noted that the calculation of the odd-numbered coefficients of the fast DCT discussed above uses 11 additions and 4 multiples beyond the initial calculation of sums and differences, where multiplies take more than one cycle – often between 4 and 11 cycles – on many microprocessors.

According to the second criterion, C2, all scaled integer basis multipliers, e.g., in the case of the FDCT, C_1 through C_7, appearing in the same sum (e.g., in the same set 1, 2, or 3 above) should possess simultaneous binary representations with the non-zero power-of-2 terms as clustered as possible; i.e., the difference between the largest power of 2 and the smallest is as small as possible. An example of a clustered representation is demonstrated by the following: $28 = 2^5 - 2^2$ and $28 = 2^4 + 2^3 + 2^2$. In the first representation, the powers of two are spread by 5 - 2 = 3, and in the second, they are spread by 4 - 2 = 2. This would make the second representation for 28 more "clustered" than the first. The advantage is this: suppose that all of the constants in a set have representations where the smallest power of 2 is 2^2. This means that there are 2 bits of precision in the calculation that can be picked up in quantization and scaling - effectively, we are dividing all of the constants by 2^2 so that we do not have to perform the shifts for the trailing zeros.

The additional magnitude represented by these trailing zeros may be reintroduced into the numbers if needed before quantization, for example, if a trailing 1 in one of the representations needs to be "picked up" for precision to make the quantization decision. (See the example under C1 above where we suggest the possible dropping of the 2^0 term.)

Fig. 4 illustrates a flow chart 300 of a method for finding the simultaneous representations for the second criteria, C2, using the condition:

$$2^n + 2^{n-1} = 2^{n+1} - 2^{n-1} \qquad (5)$$

First, "num" is set to the "repcount"-1th element of the array "reps" 310. The current count of representations for a given number is the "repcount." The variable "add" is initialized to 2 312. The bitmask is initialized to binary 0...011 314.

The bits are checked to determine if both bits in num matching the mask bits were set 340. If not 342, the mask is shifted left by one 350, effectively putting a zero at the right and the add is shifted left by one thereby increasing the power of 2 that is used for reordering in equation (5) above. If both bits in num matching the mask bits were set 344, then this bit pattern "num" is a candidate representation for doing the term reordering using the condition (5) above. Then, "num" encodes the representation by adding "add" to "num" to perform the power-of-2 change that is given by the equation shown in condition (5) above 346. This provides for more speed and storage efficiency than a brute force method; e.g., performing exhaustive search for representations, and storing all of the zero, +1, and –1 values separately. A new representation is obtained and repcount is set to repcount + 1 348. The mask is shifted left by one 350.

The shifting is performed until a predetermined maximum, "maxmask" is reached 370. If the predetermined maximum has not been reached, the routine is repeated with the new repcount so that "num" is set to the new representation 372. Otherwise 374, the routine ends.

By the way in which the representations are coded in the program, the program only keeps track of whether the coefficient in the power-of-2 polynomial (see, for example, equation (4)) is non-zero; i.e., +1 are both stored as a "1" bit, and 0 is stored as a "0" bit. The power-of-2 term in the polynomial is encoded in the bit position; e.g., bit zero (right most) corresponds to 2^0. The program does not differentiate between +1, so one might wonder how the program keeps track of representations. Since the mask scans left in the integer representation, and since the change of representation from $2^n + 2^{n-1}$ to $2^{n+1} - 2^{n-1}$ effectively moves powers of 2 left only in the polynomial representation (i.e., it only increases the power of 2 used in the representation), there is no chance of moving a power of 2 so that a power of 2 with a +1 multiplier adds with the same power of 2 with a -1 multiplier. Thus, it suffices only to keep track of whether the multipliers are non-zero, since the knowledge of the original integer, coupled with the stored integer in "reps" above, are enough to uniquely determine the representation.

As mentioned above, the present invention also works for transforms that don't split the terms nicely into disjoint sets of equations, as the FDCT (and

IDCT) could be split into sets 1, 2, and 3 above. As mentioned above, the term "disjoint", when used herein in reference to the sets of equations, means that there are no transform constants in common between equations in the two disjoint sets of equations. If these power-of-2 simultaneous representations for the constants are used, a speedup may still be achieved by choosing sets of equations in the transform, either arbitrarily or according to some cost function, and by selecting an independent scaling term for the constants in each set, grouping by powers-of-two for the transform constant representations. So one would still want to do the criteria for finding representations for the coefficients in the arbitrary equation sets. It just works out particularly nicely when the transform breaks into disjoint equation sets since the transform constants are grouped in the sets.

Below is an example with the FDCT where we use a different grouping:

$$\begin{aligned}2S(0) &= C_4(s_{0734}+s_{1625}) \\ 2S(3) &= C_3d_{07}-C_7d_{16}-C_1d_{25}-C_5d_{34} \\ 2S(5) &= C_5d_{07}-C_1d_{16}+C_7d_{25}+C_3d_{34}\end{aligned} \qquad (6)$$

$$\begin{aligned}2S(2) &= C_2d_{0734}+C_6d_{1625} \\ 2S(6) &= C_6d_{0734}-C_2d_{1625} \\ 2S(4) &= C_4(s_{0734}-s_{1625})\end{aligned} \qquad (7)$$

$$\begin{aligned}2S(1) &= C_1d_{07}+C_3d_{16}+C_5d_{25}+C_7d_{34} \\ 2S(7) &= C_7d_{07}-C_5d_{16}+C_3d_{25}-C_1d_{34}\end{aligned} \qquad (8)$$

Now one scaling term for set 6 can be obtained, and simultaneous representations for C_1, C_3, C_4, C_5, C_7 can be found for performing the three calculations in that set. Then a different scaling term for set 8 and simultaneous representations for C_1, C_3, C_5, C_7 for those equations may be found, where these representations may be very different from those representations used in the calculations for set 6.

Fig. 5 illustrates a flow chart 300 for providing faster transforms using corrective action to provide faster transform calculations and decreased execution times. At least one number resulting from an incremental calculation using transform constants in a transform is tested 310. Then, based upon the testing, when to perform a corrective action is determined 320. Once it is determined that a corrective action is to be performed, the corrective action is performed 330.

A first example of refinement occurs when each ${}_dD_k$ adds at least one additional bit of precision to the transform performed by D. A second example occurs when at least one element of the transform vector,

$\hat{F}$, is assumed to be very small, so that an entire row of ${}_dD_k$ may be approximated as zero, enabling us to skip the calculation of that at least one element of F.

In the first example, it is often the case that all of the D_k axe invertible; i.e., a matrix D_k^{-1} exists such that $D_kD_k^{-1} = D_k^{-1}D_k = I$, the identity matrix, which has ones on the upper left to lower right diagonal, and zeros elsewhere. In this case, it may be noted that

$$\hat{F}_{m+1} = D_{m+1}D_m^{-1}\hat{F}_m = \left(I + {}_dD_{m+1}D_m^{-1}\right)\hat{F}_m$$

(where I is the identity matrix); i.e., the additional step of precision is provided by performing one more step of transform to the transformed coefficients. Using this additional step transform to add the precision is the first embodiment of refinement provided by this invention, since it saves performing both IDCT and subsequent DCT: the matrix for (I + ${}_dD_{m+1}$ D_m^{-1}) can be calculated ahead of time as a matrix R_{m+1}, so that $\hat{F}_{m+1} = R_{m+1}\hat{F}_m$, a single-step transform on $\hat{F}_m$.

The second example of refinement requires a different approach. Consider a specific example, where the 2-D transform has already been performed with high precision in the first dimension, FD', ${}_dD_0$ has its 8th row zero, and ${}_dD_1$ = D – ${}_dD_0$. Then ${}_dD_0$ is not invertible; i.e., there is no way to recover the original 8 columns of FD' from ${}_dD_0$FD' (this follows from the fact that finding FD' from ${}_dD_0$FD' may be viewed as 7 equations in 8 unknowns). However, if an assumption is made for one of the 8 columns of FD', then the other 7 columns can be estimated from ${}_dD_0$FD', contingent on the assumption for the 8th column. A reasonable assumption is that the 8th column contains small elements that may be approximated as zero, since the higher-numbered transformed values tend to be less significant in real images than the lower-numbered transformed values. Then ${}_dD_0$ may be treated as an 8 x 7 matrix (ignoring the zero row), the pseudo-inverse, ${}_d\tilde{D}_o$, (as is well-known in the literature) is found by:

$${}_d\tilde{D}_o = ({}_dD'_0\;{}_dD_0)^{-1}\,{}_dD'_0,$$

with an 8th row of zeros inserted for the assumed 8th coefficients. This gives an 8 x 8 approximation for ${}_d$

D_0^{-1}, so that we can approximate

$$\hat{F}_1 = \left(I + {}_dD_1\,{}_d\tilde{D}_0\right)\hat{F}_0$$

This approximate refinement is the second embodiment of the refinement invention, which saves the cycles of the IDCT followed by the DCT, as in the first example.

The abort procedure is used to determine when a calculation can be terminated before its completion to save cycles, when the result of the calculation is projected to be too small, so that it wilt be quantized to zero. One example of the application of the abort procedure appears in the example above, where at least one low-magnitude transform coefficient is not calculated, being essentially equivalent to setting the corresponding row or rows of the transform matrix to zero. Another example is stopping a calculation with limited precision, when additional transform precision is projected to provide negligible additional information in the transformed values; e.g., when the result of the calculation is projected to be small. An alternative method involves testing the magnitude of the sums and/or differences of some of the inputs to the transform. For example, for the FDCT, the following equation calculates the second transform coefficient: $2S(2)=C_2 d_{0734}+C_6 d_{162}$

where

$d_{0734}=s_{07}-s_{34}$ and $d_{1625}=s_{16}-s_{25}$, notation from Pennebaker and Mitchell's JPEG text. The magnitudes of these values can be tested for impact on subsequent processing of the transform coefficients. In this example, if S(2) is less than the magnitude of Q/2 (where Q is a quantization value for S(2)), then S(2) will be quantized to zero. This translates to a test of whether d_{073} is less than $Q/(2C_2)$ in magnitude and d_{162} is less than $Q/(2C_6)$ in magnitude. If this test is met, then the calculation for S(2) can be aborted, and S(2) set to its quantized value of zero. This method of testing sums and/or differences of the input values can be extended to all of the equations for the FDCT.

It is not obvious how to turn a comparison such as $-\hat{T}<\hat{F}<\hat{T}$ (a term-by-term range check for the members of vector or matrix $\hat{F}$), where the elements of $\hat{T}$ are all non-negative, into a term-by-term comparison of the elements of F, -T < F < T, where the elements of T are all non-negative, and where satisfying the test on F is sufficient to satisfy the test on $\hat{F}$. The difficulty arises from the fact that the DCT employs both positive and negative operations, which destroys the term-by-term ordering in the equation. Specifically, it cannot be said that $-\hat{T}<\hat{F}<\hat{T}$ implies that $-D^{-1}\hat{T}<F<D^{-1}\hat{T}$.

Thus, the abort involves terminating the precision of an operation when additional transform precision is projected to have an acceptable or negligible effect on the results of the subsequent processing operations, e.g., quantization or comparison. For example, the coefficients of the DCT can be scaled by an integer and approximated as sums of powers of 2. For the odd terms, one of these approximations is as follows:

$$41D=41\begin{bmatrix}C_1 & C_3 & C_5 & C_7\\ C_3 & -C_7 & -C_1 & -C_5\\ C_5 & -C_1 & C_7 & C_3\\ C_7 & -C_5 & C_3 & -C_1\end{bmatrix}\approx 32\begin{bmatrix}1 & 1 & 1 & 0\\ 1 & 0 & -1 & -1\\ 1 & -1 & 0 & 1\\ 0 & -1 & 1 & -1\end{bmatrix}+8\begin{bmatrix}1 & 0 & -1 & 1\\ 0 & -1 & -1 & 1\\ -1 & -1 & 1 & 0\\ 1 & 1 & 0 & -1\end{bmatrix}+2\begin{bmatrix}0 & 1 & -1 & 0\\ 1 & 0 & 0 & 1\\ -1 & 0 & 0 & 1\\ 0 & 1 & 1 & 0\end{bmatrix}+\begin{bmatrix}0 & 0 & -1 & 0\\ 0 & 0 & 0 & 1\\ -1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\end{bmatrix}$$

which we write as (using the notation from above),

$$41D\approx 2^5\,{}_dD_0+2^3\,{}_dD_1+2^1\,{}_dD_2+2^0\,{}_dD_3$$

Also, as mentioned above, all of the matrices above, and their sequential sums, are invertible. Now, if $\left|j\hat{F}\right|<<$, i.e., the jth element of $\hat{F}$ is very small in magnitude, then j (32 ${}_dD_0$ F) should be small. If it is not, then j ((8 ${}_dD_1$ + 2 ${}_dD_2$ + ${}_dD_3$) F) will not be able to cancel it out to make the final result small. The relative magnitudes of the results of the calculations may be checked. If one of the transform values, for one of the intermediate precisions, is small compared to the other values, or is small compared to some pre-determined threshold, then subsequent refinements for that transform value can be aborted.

Fig. 6 illustrates a flow chart 400 of the abort method according to the present invention that demonstrates aborting further iterations of the transform coefficient calculation process. In Fig. 6, at least one incrementally calculated number is tested 410. If certain criteria are met, further calculations are aborted 420. The incremental calculation of a transform coefficient may be aborted when an error resulting from terminating the incremental calculation is acceptable. For example, the incremental calculation may be terminated when a determination is made that the incremental calculation will result in a number that is projected to be within a predetermined range, e.g., a transform coefficient that does satisfy a precision requirement. Alternatively, the incremental calculation of the transform coefficient may be aborted when a transform coefficient is going to be within a predetermined range of zero.

Fig. 7 is a flow chart 500 of the testing of the at least one incrementally calculated number. In Fig. 7, the incremental calculation is tested to determine when a refinement to the transform coefficient will not change the result 510. This testing may be carried out in at least two ways as shown in Fig. 7. A refinement to the transform coefficient may be determined not to change the result when, after checking the relative magnitudes of the results of the incremental calculations, an intermediate calculation of at least one transform coefficient is small compared to the intermediate calculation of another transform coefficient 520. Alternatively, a refinement to the transform coefficient may be determined not to change the result when, after checking the magnitude of the results of at least one incremental calculation, at least one intermediate calculation of the transform coefficient is less than a predetermined threshold 530.

Fig. 8 is a flow chart 600 of the refinement method according to the present invention. First, a determination is made whether the transform requires more precision 610. The transform is a transform matrix, wherein a refinement matrix may be used to increase precision of the incremental calculation of the transform coefficients. When more precision is required, a refinement matrix is applied to the transform 620. The refinement matrix is generated offline or at initialization and is based on approximately calculated transform constants.

Fig. 9 illustrates a flow chart 700 of a first method for generating a refinement matrix. First, it is recognized that an approximate transform is invertible 710. The refinement matrix given by $I+{}_dD_{m+1}\,D_m^{-1}$ is generated 720. Then, the transform is structured for efficient computation 730.

However, as described above, when ${}_dD_0$ is not invertible; there is no way to recover the original 8 columns of FD' from ${}_dD_0$FD'. Fig. 10 is a flow chart 800 showing a second method for generating a refinement matrix when ${}_dD_0$ is not invertible. It is first recognized that recovery of the nth column of a transform matrix for generating the transform is impossible 810. A pseudo inverse for a portion of the transform matrix is calculated 820. Then, an approximation for the refinement matrix is generated using the pseudo inverse for the transform matrix 830. The approximation of the refinement matrix comprises $I+{}_dD_1\,{}_d\tilde{D}_0$.

Fig. 11 illustrates a block diagram 400 of a printer 410 according to the present invention. In Fig. 11, the printer 410 receives image data 412 from a host processor 410. The image data 412 is provided into memory 430 where the image data may be arranged into 8x8 block samples. The 8x8 block samples are then processed by a processor 440, such as a raster image processor. The raster image processor 440 provides a compressed print stream representing the image data to a printhead driving circuit 450. The printhead driving circuit 450 then controls the printhead 460 to generate a printout 470 of the image data.

The process illustrated with reference to Figs. 1-4 may be tangibly embodied in a computer-readable medium or carrier 490, e.g. one or more of the fixed and/or removable data storage devices illustrated in Fig. 11, or other data storage or data communications devices. The computer program may be loaded into the memory 492 to configure the processor 440 of Fig. 11, for execution. The computer program comprises instructions which, when read and executed by the processor 440 of Fig. 11, causes the processor 440 to perform the steps necessary to execute the steps or elements of the present invention.

Fig. 12 illustrates a data analyzing system 500 according to the present invention. In Fig. 12, a transform 510 receives a block of data 512 to be analyzed. The transform 510 uses transform equations 520 to generate transformed data 524. Transform equations 520 are split into at least one sub-transform having at least two transform constants. The at least two transform constants for each collection are independently scaled with a scaling term to maintain a substantially uniform ratio between the at least two transform constants within the at least one collection, wherein the scaling term may be chosen according to a predetermined cost function. The transformed data 524 may then be optionally quantized by quantizer 530. The quantization values in quantizer 530 are adjusted to reflect the scaling terms used for each coefficient.

Fig. 13 illustrates another data analyzing system 600 according to the present invention. In Fig. 13, a transform 610 receives a block of data 612 to be analyzed. The transform 610 uses transform equations 620 to generate transformed data 624. Transform equations 620 are split into at least one sub-transform having at least two transform constants. The at least two transform constants for each collection are independently scaled with a scaling term to maintain a substantially uniform ratio between the at least two transform constants within the at least one collection, wherein the scaling term may be chosen according to a predetermined cost function. The transformed data 624 may then be compared to scaled comparison values in comparator 630.

The foregoing description of the exemplary embodiment of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not with this detailed description, but rather by the claims appended hereto.

Claims:

1. A method comprising the steps of:

arranging transform equations into at least one collection having at least two transform constants; and

independently scaling the at least two transform constants for each collection with a scaling term to maintain a substantially uniform ratio between the at least two transform constants within the at least one collection.

2. A method according to Claim 1 wherein the scaling term is one of the transform constants from the at least one collection.

3. A method according to Claim 1 wherein each of the transform constant is represented by an estimated transform constant approximated by sums of powers of 2.

4. A method according to Claim 1 further comprising separating data into at least one block and transforming the block into transform data via the equations applied to maintain a substantially uniform ratio between the at least two transform constants within the at least one collection.

5. A method according to one of Claims 1 and 4 wherein the scaling terms is chosen according to a predetermined cost function.

6. A method according to Claim 5 further comprising the step of determining the predetermined cost function by selecting the scaling term such that the largest error on any transform coefficient is no larger than a predetermined error percentage.

7. A method according to Claim 5 further comprising the step of determining the predetermined cost function by selecting the scaling term such that predetermined transform constant have an error less than or equal to a predetermined error percentage.

8. A method according to Claim 5 further comprising the step of determining the predetermined cost function by selecting the scaling term and reprsentations for the transform constant such that all transform constant for a collection posses simultaneous binary representations with predetermined characteristics.

9. A method according to Claim 8 wherein the selecting of the scaling term and representations for the transform constants so that all transform constants for a collection possess simultaneous binary representations with a minimum number of common power-of-2 terms is implemented when binary arithmetic shifts may be more efficient than multiplication operations.

10. A method according to claim 9 wherein all representations of the scaled constants are found by the steps of:

a) setting a first variable to an ith element in the block;

b) initializing a second variable to a value of 2;

c) initializing a bitmask to binary 3;

d) analyzing the bits to determine whether the ith element indicated by the first variable is a candidate representation for doing the term reordering using $2^n + 2^{n-1} = 2^{n+1} - 2^{n-1}$;

e) encoding the ith element by adding the second variable to the first variable to perform an effective power-of-2 change given by $2^n + 2^{n-1} = 2^{n+1} - 2^{n-1}$;

f) obtaining a new representation and incrementing the first variable to the i+1th element;

g) shifting the bitmask and second variable left one bit; and

h) repeating steps d-g.

11. The method of claim 10 further comprising the steps of: shifting the bitmask left after checking if the first variable matching the mask bits were set thereby putting a zero at the right and increasing the power of 2 that is used for reordering in $2^n + 2^{n-1} = 2^{n+1} - 2^{n-1}$.

12. The method of claim 9 wherein the predetermined characteristics comprise a maximized clustering of non-zero power-of-2 terms.

13. The method of claim 12 wherein the step of selecting of the scaling term so that all transform constants for a collection possess simultaneous binary representations with a maximized clustering of non-zero power-of-2 terms is implemented when multiplication operations employing smaller integers are more desirable than multiplies employing larger numbers.

14. The method of claim 13 wherein whether the coefficient in a power-of-2 polynomial representing the constant is non-zero is tracked.

15. The method of claim 14 wherein a value of the bit position determines the power-of-2 term.

16. The method of claim 13 wherein maximizing the clustering of non- zero power-of-2 terms comprises finding all representations of the scaled constants by the steps of:

a) setting a first variable to an ith element in the block;

b) initializing a second variable to a value of 2;

c) initializing a bitmask to binary 3;

d) analyzing the bits to determine whether the ith element indicated by the first variable is a candidate representation for doing the term reordering using $2^n + 2^{n-1} = 2^{n+1} - 2^{n-1}$;

e) encoding the ith element by adding the second variable to the first variable to perform an effective power-of-2 change given by $2^n + 2^{n-1} = 2^{n+1} - 2^{n-1}$;

f) obtaining a new representation and incrementing the first variable to the i+1th element;

g) shifting the bitmask and second variable left one bit; and

h) repeating steps d-g.

17. The method of claim 16 further comprising: shifting the bitmask left after checking if the first variable matching the mask bits were set thereby putting a zero at the right and increasing the power of 2 that is used for reordering in $2^n + 2^{n-1} = 2^{n+1} - 2^{n-1}$.

18. The method of claim 1 wherein the collections represent disjoint sets of transform equations of partial calculations.

19. The method of claim 1 wherein the collections do not represent disjoint sets of transform equations of partial calculations.

20. The method of claim 19 further comprising selecting an independent scaling term for the transform constants in each of the at least one collection.

21. A method according to Claim 1 further comprising the steps of:

testing at least one number resulting from an incremental calculation of transform coefficients during a transform;

determining whether to perform a corrective action based upon the testing; and

performing the corrective action when a corrective action is determined to be needed.

22. A method according to claim 21 wherein the step of determining whether to perform a corrective action comprises detecting whether the incremental calculation of the transform coefficients will result in transform coefficients with unacceptable precision and the performing corrective action comprises refining the at least one number.

23. A method according to claim 22 wherein the transform comprises a transform matrix and wherein the refining comprises applying a refinement matrix for increasing precision of the incremental calculation of the transform constants.

24. A method according to claim 23 wherein the refinement matrix comprises $I + {}_{d}D_{m+1}\, D_{m}^{-1}$.

25. A method according to claim 21 further comprising the step of generating at least one refinement matrix based on approximately calculated transform constants.

26. A method according to claim 25 wherein the step of generating at least one refinement matrix is performed offline or at initialization.

27. A method according to claim 25 wherein the step of generating the at least one refinement matrix comprises recognizing that an approximate transform is invertible, generating the refinement matrix given by $I+_{d}D_{m+1}\ D_{m}^{-1}$, and structuring the transform for efficient computation.

28. A method according to claim 25 wherein the step of generating the at least one refinement matrix comprises the steps of:

recognizing that recovery of the nth column of a transform matrix for generating the transform is impossible;

calculating a pseudo inverse for a portion of the transform matrix; and

generating an approximation for the at least one refinement matrix using the pseudo inverse for the transform matrix.

29. A method according to claim 28 wherein the approximation of the refinement matrix comprises $I + {}_{d}D_{1}\,{}_{d}\widetilde{D}_{0}$.

30. A method according to claim 21 wherein the step of determining whether to perform a corrective action further comprises determining whether an error resulting from terminating the incremental calculation is acceptable and the performing corrective action comprises aborting the incremental calculation of a transform coefficient.

31. A method according to claim 30 wherein the incremental calculation is terminated when a determination is made that the incremental calculation will result in a number that is projected to be within a predetermined range.

32. A method according to claim 31 wherein the number that is projected to be within a predetermined range comprises a transform coefficient that does satisfy a precision requirement.

33. A method according to claim 31 wherein the incremental calculation is terminated when a refinement to the transform coefficient is determined not to change the result.

34. A method according to claim 33 wherein a refinement to the transform coefficient is determined not to change the result when, after checking the relative magnitudes of the results of the incremental calculations, an intermediate calculation of at least one transform coefficient is small compared to the intermediate calculation of another transform coefficient.

35. A method according to claim 33 wherein a refinement to the transform coefficient is determined not to change the result when, after checking the magnitude of the results of at least one incremental calculation, at least one intermediate calculation of the transform coefficient is less than a predetermined threshold.

36. A method according to claim 21 wherein the determining further comprises determining that a transform coefficient is going to be within a predetermined range of zero and the performing corrective action comprises aborting the incremental calculation of the transform coefficient.

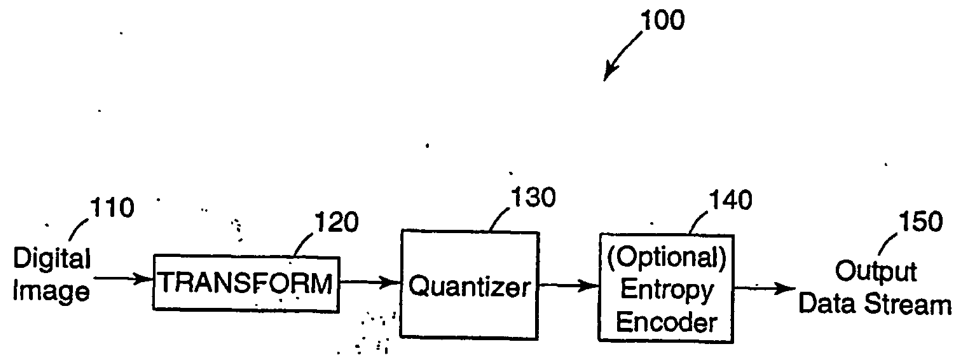

100
110
Digital Image
120
TRANSFORM
130
Quantizer
140
(Optional) Entropy Encoder
150
Output Data Stream

Fig. 1

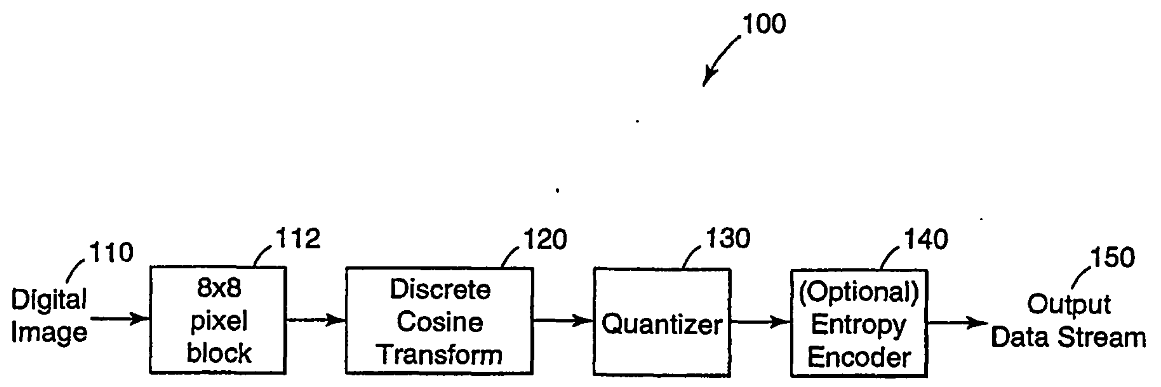

100
110
Digital Image
112
8x8 pixel block
120
Discrete Cosine Transform
130
Quantizer
140
(Optional) Entropy Encoder
150
Output Data Stream

Fig. 2

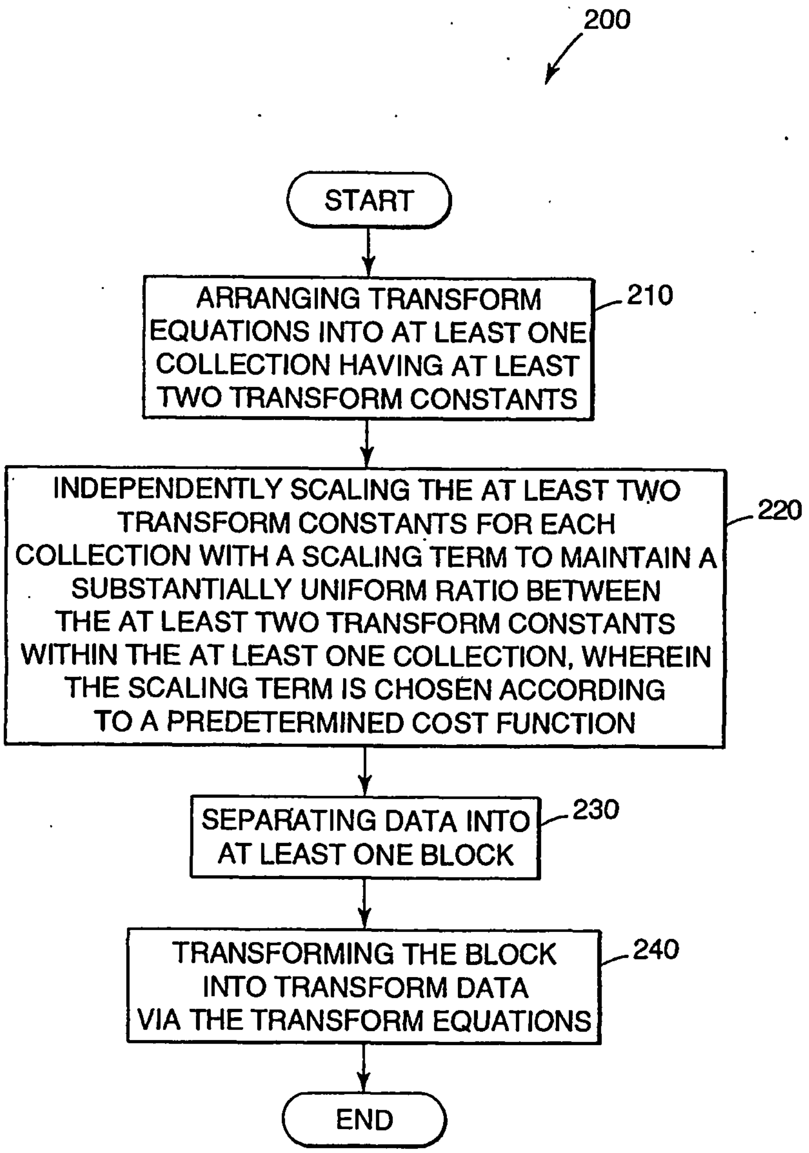

200
START
ARRANGING TRANSFORM EQUATIONS INTO AT LEAST ONE COLLECTION HAVING AT LEAST TWO TRANSFORM CONSTANTS
210
INDEPENDENTLY SCALING THE AT LEAST TWO TRANSFORM CONSTANTS FOR EACH COLLECTION WITH A SCALING TERM TO MAINTAIN A SUBSTANTIALLY UNIFORM RATIO BETWEEN THE AT LEAST TWO TRANSFORM CONSTANTS WITHIN THE AT LEAST ONE COLLECTION, WHEREIN THE SCALING TERM IS CHOSEN ACCORDING TO A PREDETERMINED COST FUNCTION
220
SEPARATING DATA INTO AT LEAST ONE BLOCK
230
TRANSFORMING THE BLOCK INTO TRANSFORM DATA VIA THE TRANSFORM EQUATIONS
240
END

Fig. 3

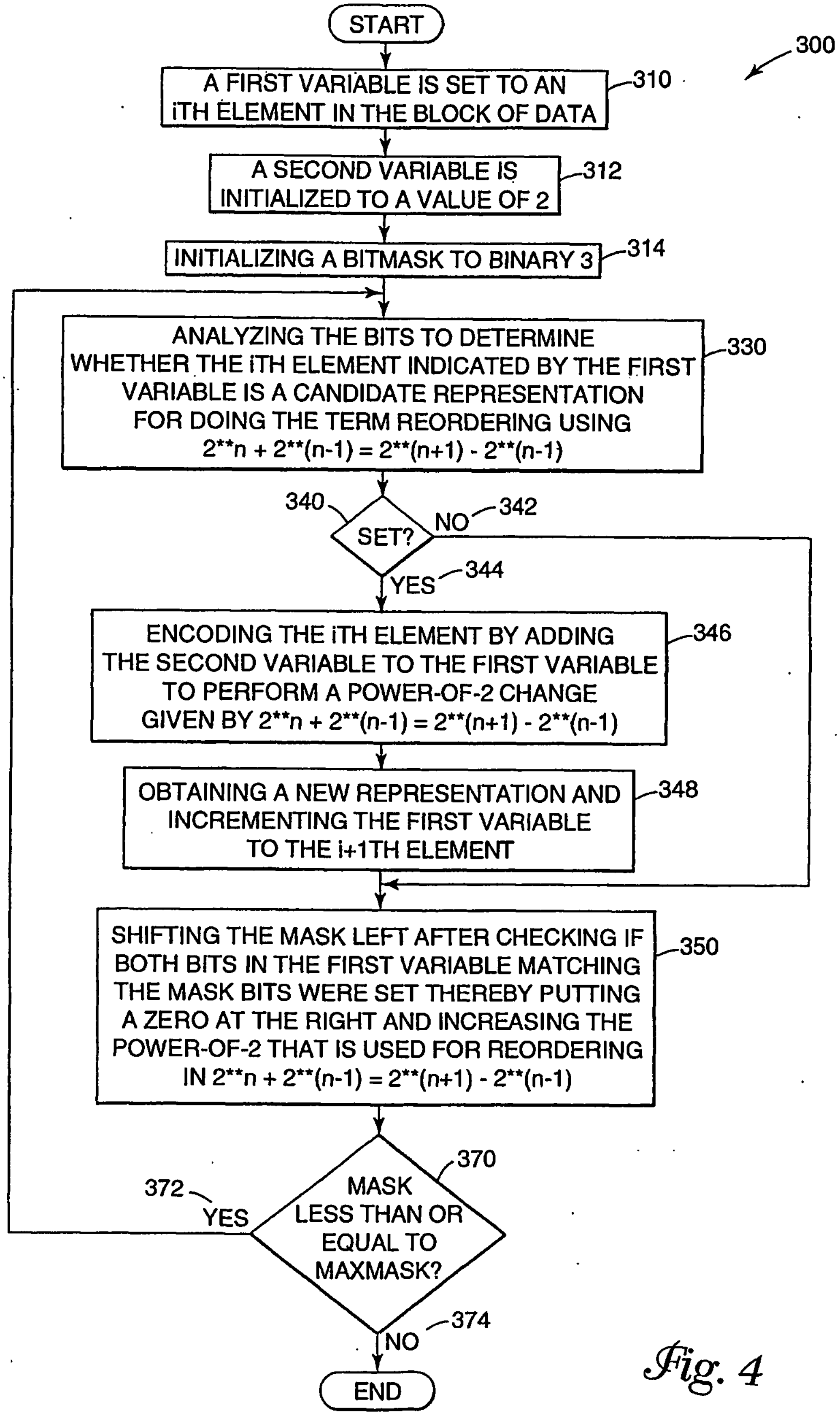

START
300
A FIRST VARIABLE IS SET TO AN iTH ELEMENT IN THE BLOCK OF DATA
310
A SECOND VARIABLE IS INITIALIZED TO A VALUE OF 2
312
INITIALIZING A BITMASK TO BINARY 3
314
ANALYZING THE BITS TO DETERMINE WHETHER THE iTH ELEMENT INDICATED BY THE FIRST VARIABLE IS A CANDIDATE REPRESENTATION FOR DOING THE TERM REORDERING USING 2**n + 2**(n-1) = 2**(n+1) - 2**(n-1)
330
340
SET?
NO
342
YES
344
ENCODING THE iTH ELEMENT BY ADDING THE SECOND VARIABLE TO THE FIRST VARIABLE TO PERFORM A POWER-OF-2 CHANGE GIVEN BY 2**n + 2**(n-1) = 2**(n+1) - 2**(n-1)
346
OBTAINING A NEW REPRESENTATION AND INCREMENTING THE FIRST VARIABLE TO THE i+1TH ELEMENT
348
SHIFTING THE MASK LEFT AFTER CHECKING IF BOTH BITS IN THE FIRST VARIABLE MATCHING THE MASK BITS WERE SET THEREBY PUTTING A ZERO AT THE RIGHT AND INCREASING THE POWER-OF-2 THAT IS USED FOR REORDERING IN 2**n + 2**(n-1) = 2**(n+1) - 2**(n-1)
350
372
YES
MASK LESS THAN OR EQUAL TO MAXMASK?
370
NO
374
END

Fig. 4

Fig. 5

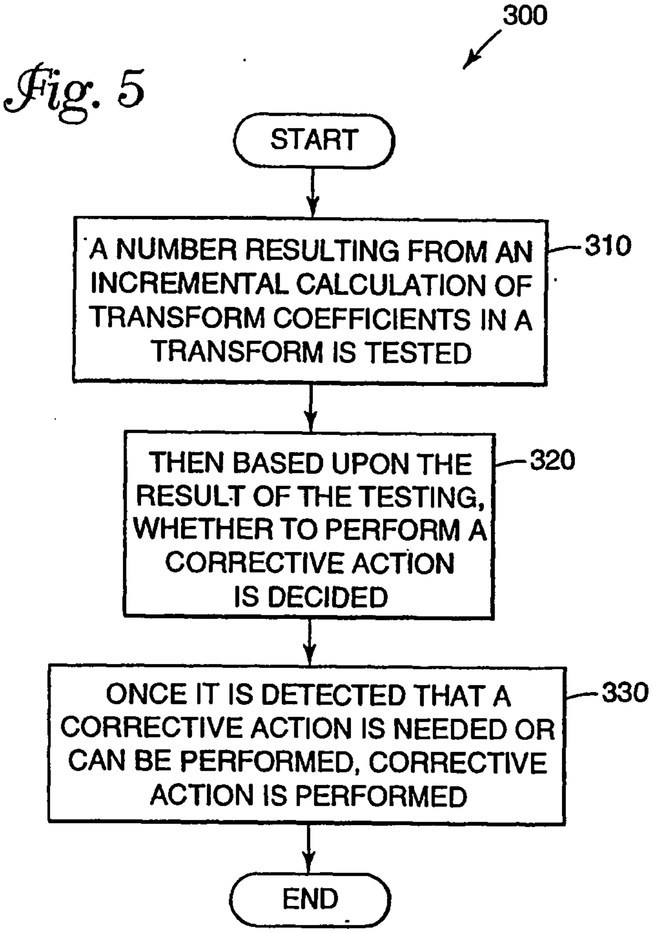
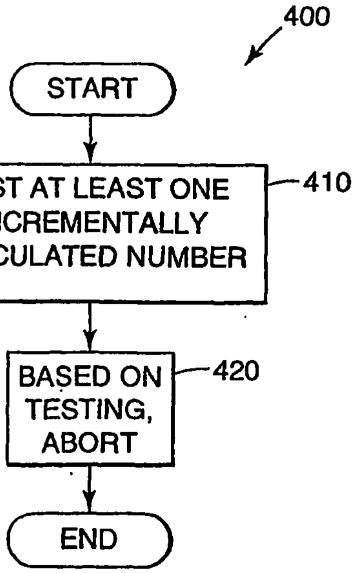

300
START
A NUMBER RESULTING FROM AN INCREMENTAL CALCULATION OF TRANSFORM COEFFICIENTS IN A TRANSFORM IS TESTED
310
THEN BASED UPON THE RESULT OF THE TESTING, WHETHER TO PERFORM A CORRECTIVE ACTION IS DECIDED
320
ONCE IT IS DETECTED THAT A CORRECTIVE ACTION IS NEEDED OR CAN BE PERFORMED, CORRECTIVE ACTION IS PERFORMED
330
END

Fig. 6


400
START
TEST AT LEAST ONE INCREMENTALLY CALCULATED NUMBER
410
BASED ON TESTING, ABORT
420
END

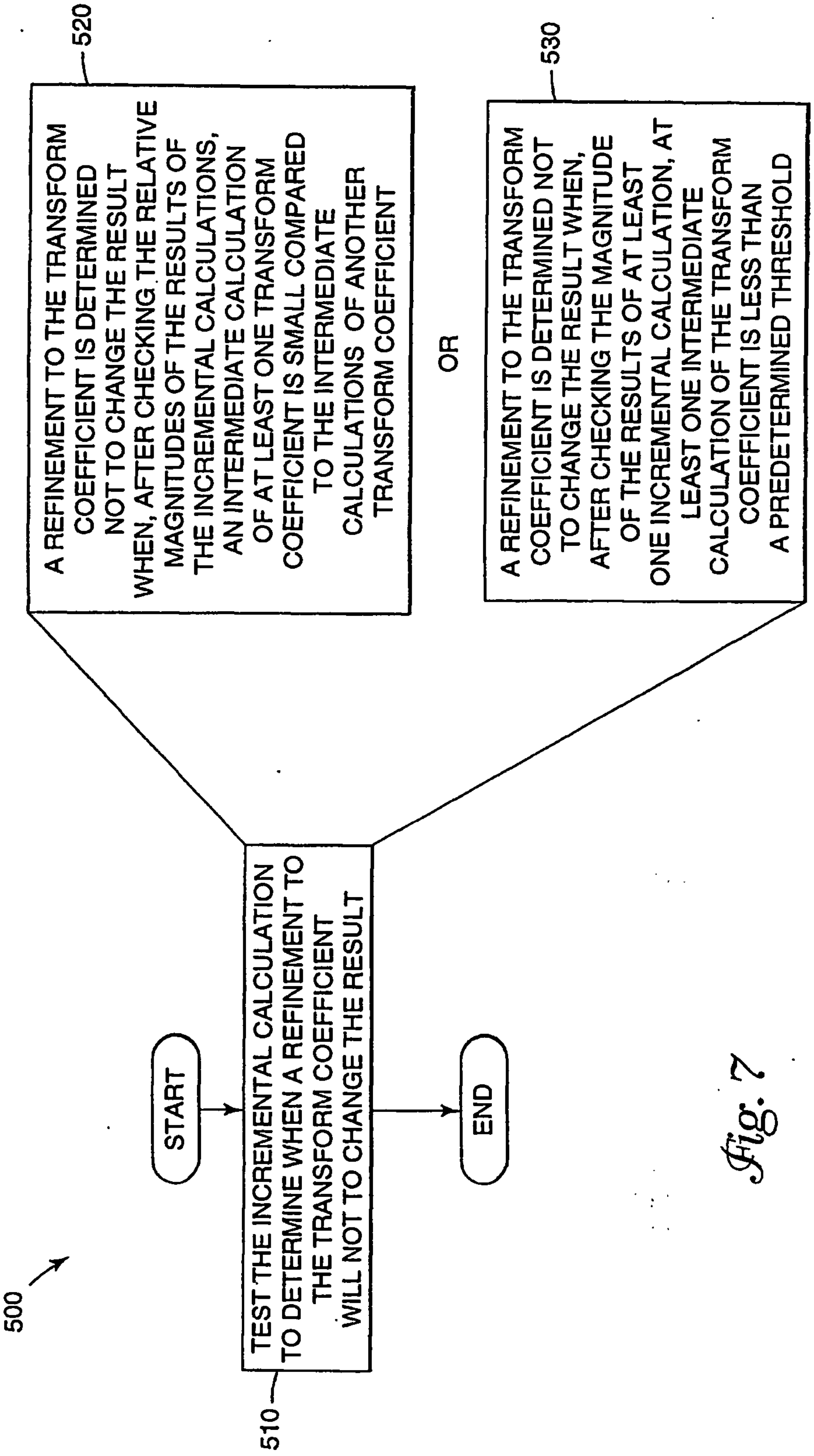

500
START
510
TEST THE INCREMENTAL CALCULATION TO DETERMINE WHEN A REFINEMENT TO THE TRANSFORM COEFFICIENT WILL NOT TO CHANGE THE RESULT
END
520
A REFINEMENT TO THE TRANSFORM COEFFICIENT IS DETERMINED NOT TO CHANGE THE RESULT WHEN, AFTER CHECKING THE RELATIVE MAGNITUDES OF THE RESULTS OF THE INCREMENTAL CALCULATIONS, AN INTERMEDIATE CALCULATION OF AT LEAST ONE TRANSFORM COEFFICIENT IS SMALL COMPARED TO THE INTERMEDIATE CALCULATIONS OF ANOTHER TRANSFORM COEFFICIENT
OR
530
A REFINEMENT TO THE TRANSFORM COEFFICIENT IS DETERMINED NOT TO CHANGE THE RESULT WHEN, AFTER CHECKING THE MAGNITUDE OF THE RESULTS OF AT LEAST ONE INCREMENTAL CALCULATION, AT LEAST ONE INTERMEDIATE CALCULATION OF THE TRANSFORM COEFFICIENT IS LESS THAN A PREDETERMINED THRESHOLD

Fig. 7

Fig. 8

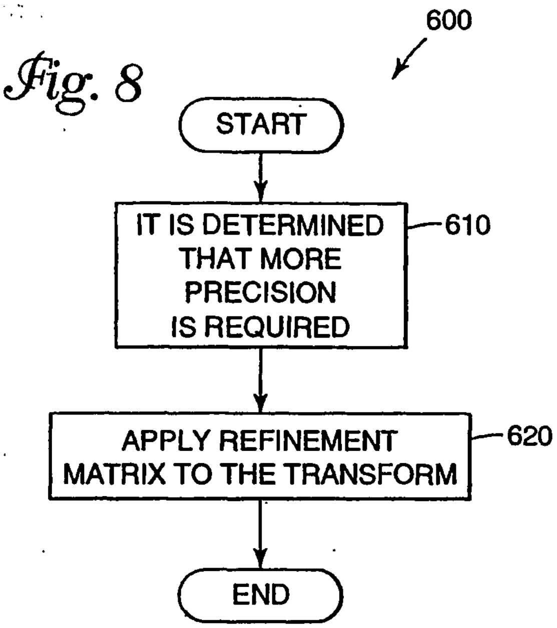

600
START
IT IS DETERMINED THAT MORE PRECISION IS REQUIRED
610
APPLY REFINEMENT MATRIX TO THE TRANSFORM
620
END

Fig. 9

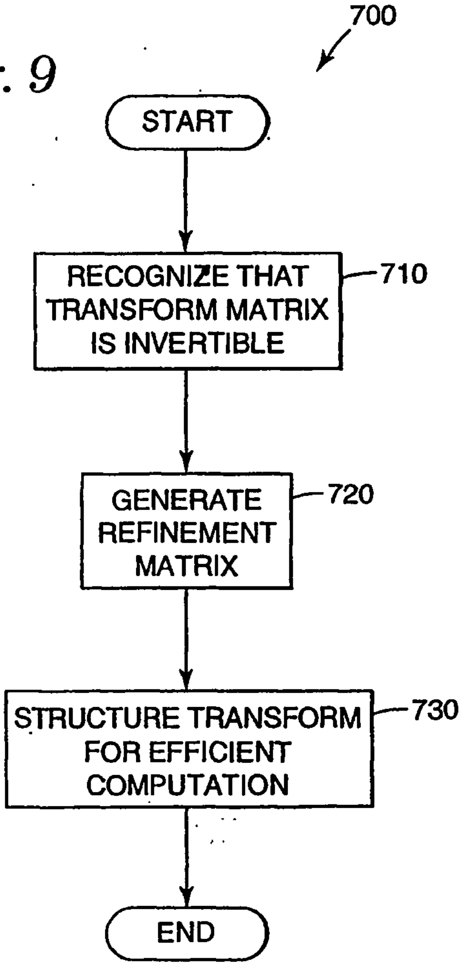
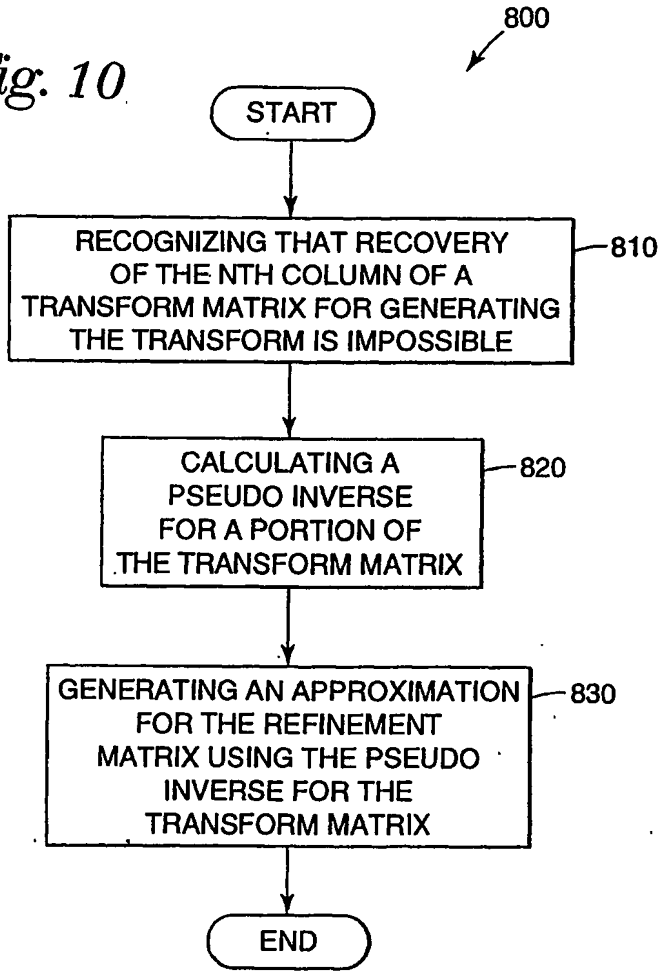

700
START
RECOGNIZE THAT TRANSFORM MATRIX IS INVERTIBLE
710
GENERATE REFINEMENT MATRIX
720
STRUCTURE TRANSFORM FOR EFFICIENT COMPUTATION
730
END

Fig. 10


800
START
RECOGNIZING THAT RECOVERY OF THE NTH COLUMN OF A TRANSFORM MATRIX FOR GENERATING THE TRANSFORM IS IMPOSSIBLE
810
CALCULATING A PSEUDO INVERSE FOR A PORTION OF THE TRANSFORM MATRIX
820
GENERATING AN APPROXIMATION FOR THE REFINEMENT MATRIX USING THE PSEUDO INVERSE FOR THE TRANSFORM MATRIX
830
END

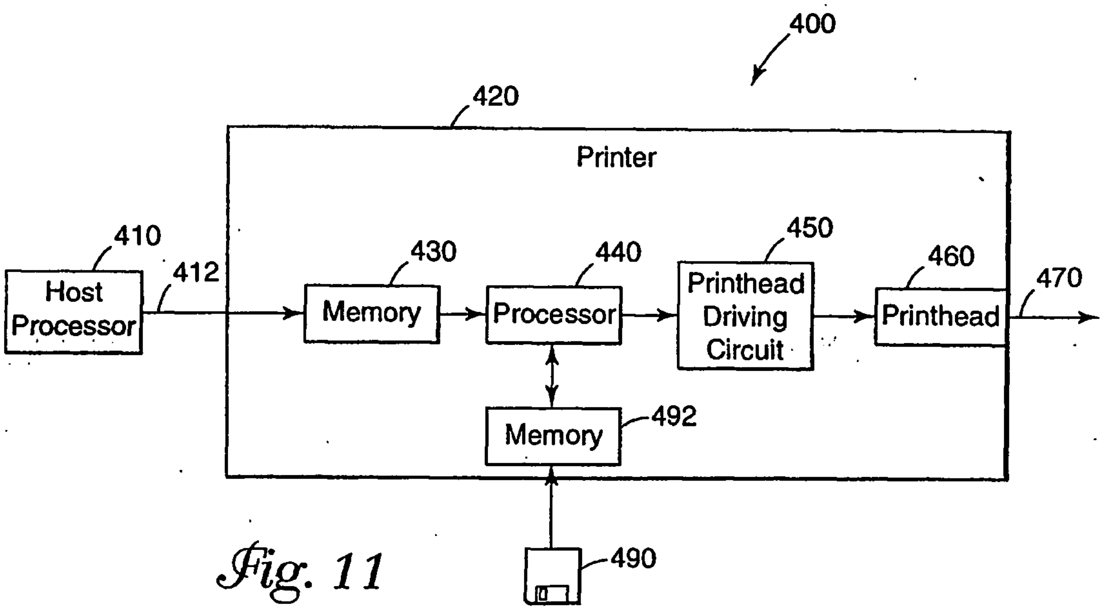

400
420
Printer
410
Host Processor
412
430
Memory
440
Processor
450
Printhead Driving Circuit
460
Printhead
470
Memory
492
490

Fig. 11

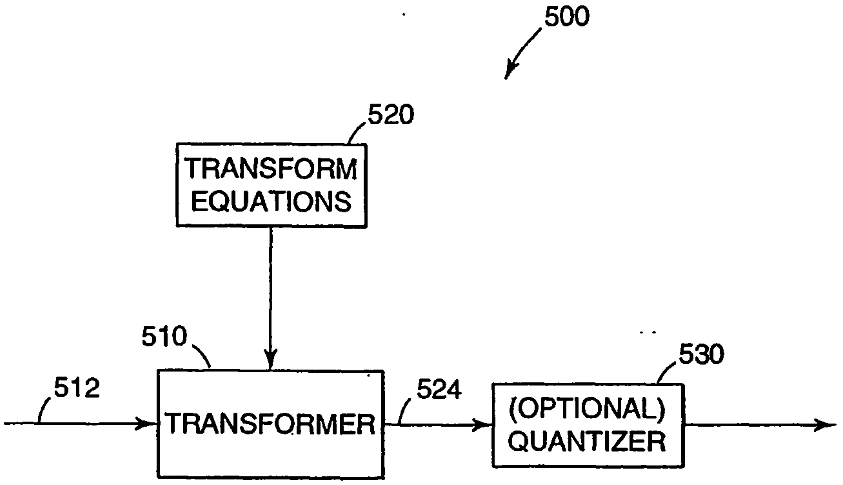

500
520
TRANSFORM EQUATIONS
510
512
TRANSFORMER
524
530
(OPTIONAL) QUANTIZER

Fig. 12

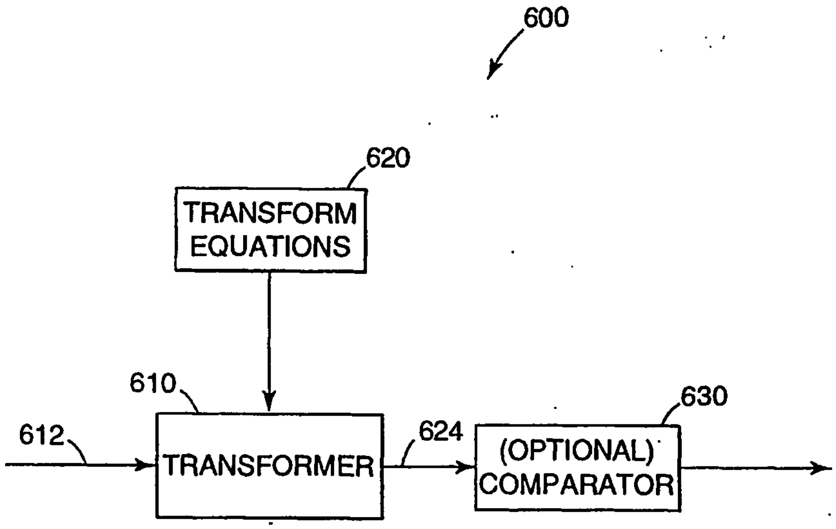

600
620
TRANSFORM EQUATIONS
610
612
TRANSFORMER
624
630
(OPTIONAL) COMPARATOR

Fig. 13

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/27778

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :G06F 17/14
US CL :708/400
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 708/400,401,402,403,404,405,406

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EAST
search terms: transform, cosine, scale, power of two

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X --- Y	US 5,854,757 A (DIERKE) 29 December 1998, cols. 3-7.	1-9,12-15 and 18-20. --- 21-23,25,26 and 30-36.
A	US 5,394,349 A (EDDY) 28 February 1995, figures 4.	1-36.
A	US 4,849,922 A (RIOLFO) 18 July 1989, cols. 8-10.	1-36.

□ Further documents are listed in the continuation of Box C. □ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance		
"E"	earlier document published on or after the international filing date	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed	"&"	document member of the same patent family

Date of the actual completion of the international search 16 JANUARY 2002	Date of mailing of the international search report 28 JAN 2002
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Authorized officer CHUONG D. NGO
Facsimile No. (703) 305-3230	Telephone No. (703) 305-3800